Programming Fundamentals
Lecture #18
File Handling in C Programming

Junaid Hassan

Lecturer CS & IT Department UOS MBDIN

junaidte14@gmail.com

Introduction:

- Storage of data in variables and arrays is temporary—such data is lost when a program terminates.
- Files are used for permanent retention of data.
- Computers store files on secondary storage devices, especially disk storage devices.
- In this lecture, we explain how data files are created, updated and processed by C programs

Data Hierarchy:

- Ultimately, all data items processed by a computer are reduced to combinations of zeros and ones.
- This occurs because it's simple and economical to build electronic devices that can assume two stable states—one of the states represents 0 and the other represents 1.
- The smallest data item in a computer can assume the value 0 or the value 1. Such a data item is called a bit (short for "binary digit"—a digit that can assume one of two values)

Data Hierarchy:

- It's difficult to work with data in the low-level form of bits. Instead, programmers prefer to work with data in the form of decimal digits (i.e., 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9), letters (i.e., A–Z, and a–z), and special symbols (i.e., $, @, %, &, *, (,), -, +, ", :, ?, /, and others).
- Digits, letters, and special symbols are referred to as characters. The set of all characters that may be used to write programs and represent data items on a particular computer is called that computer's character set.

Data Hierarchy:

- Since computers can process only 1′s and 0′s, every character in a computer's character set is represented as a pattern of 1′s and 0′s (called a byte). Today, bytes are most commonly composed of eight bits.
- You create programs and data items as characters; computers then manipulate and process these characters as patterns of bits

Files in C Programming:

- A file is a group of related records. A company's payroll file normally contains one record for each employee. Thus, a payroll file for a small company might contain only 22 records, whereas a payroll file for a large company might contain 100,000 records.
- It's not unusual for an organization to have hundreds or even thousands of files, with some containing billions or even trillions of characters of information

Files and Streams:

- C views each file simply as a sequential stream of bytes. Each file ends either with an end-of-file marker or at a specific byte number recorded in a system-maintained, administrative data structure.
- When a file is opened, a stream is associated with the file. Three files and their associated streams are automatically opened when program execution begins—the standard input, the standard output and the standard error.

Files and Streams:

- Streams provide communication channels between files and programs. For example, the standard input stream enables a program to read data from the keyboard, and the standard output stream enables a program to print data on the screen.

- Opening a file returns a pointer to a FILE structure (defined in <stdio.h>) that contains information used to process the file

Files and Streams:

- The standard library provides many functions for reading data from files and for writing data to files.
- Function fgetc, like getchar, reads one character from a file. Function fgetc receives as an argument a FILE pointer for the file from which a character will be read. The call fgetc(stdin) reads one character from stdin—the standard input. This call is equivalent to the call getchar().

Files and Streams

- Function fputc, like putchar, writes one character to a file. Function fputc receives as arguments a character to be written and a pointer for the file to which the character will be written. The function call fputc('a', stdout) writes the character 'a' to stdout—the standard output. This call is equivalent to putchar('a')

Creating a Sequential-Access File

```c
/* Fig. 11.3: fig11_03.c
   Create a sequential file */
#include <stdio.h>

int main( void )
{
   int account; /* account number */
   char name[ 30 ]; /* account name */
   double balance; /* account balance */

   FILE *cfPtr; /* cfPtr = clients.dat file pointer */

   /* fopen opens file. Exit program if unable to create file  */
   if ( ( cfPtr = fopen( "clients.dat", "w" ) ) == NULL ) {
      printf( "File could not be opened\n" );
   } /* end if */
   else {
      printf( "Enter the account, name, and balance.\n" );
      printf( "Enter EOF to end input.\n" );
      printf( "? " );
      scanf( "%d%s%lf", &account, name, &balance );

      /* write account, name and balance into file with fprintf */
      while ( !feof( stdin ) ) {
         fprintf( cfPtr, "%d %s %.2f\n", account, name, balance );
         printf( "? " );
         scanf( "%d%s%lf", &account, name, &balance );
      } /* end while */
```

Creating a Sequential-Access File

```c

      fclose( cfPtr ); /* fclose closes file */
   } /* end else */

   return 0; /* indicates successful termination */
} /* end main */
```

```
Enter the account, name, and balance.
Enter EOF to end input.
? 100 Jones 24.98
? 200 Doe 345.67
? 300 White 0.00
? 400 Stone -42.16
? 500 Rich 224.62
? ^Z
```

Reading Data from a Sequential-Access File

```c
/* Fig. 11.7: fig11_07.c
   Reading and printing a sequential file */
#include <stdio.h>

int main( void )
{
   int account;     /* account number */
   char name[ 30 ]; /* account name */
   double balance;  /* account balance */

   FILE *cfPtr;     /* cfPtr = clients.dat file pointer */

   /* fopen opens file; exits program if file cannot be opened */
   if ( ( cfPtr = fopen( "clients.dat", "r" ) ) == NULL ) {
      printf( "File could not be opened\n" );
   } /* end if */
   else { /* read account, name and balance from file */
      printf( "%-10s%-13s%s\n", "Account", "Name", "Balance" );
      fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
```

Reading Data from a Sequential-Access File

```c

      /* while not end of file */
      while ( !feof( cfPtr ) ) {
         printf( "%-10d%-13s%7.2f\n", account, name, balance );
         fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
      } /* end while */

      fclose( cfPtr ); /* fclose closes the file */
   } /* end else */

   return 0; /* indicates successful termination */
} /* end main */
```

```
Account   Name         Balance
100       Jones          24.98
200       Doe           345.67
300       White           0.00
400       Stone         -42.16
500       Rich          224.62
```